

10 A

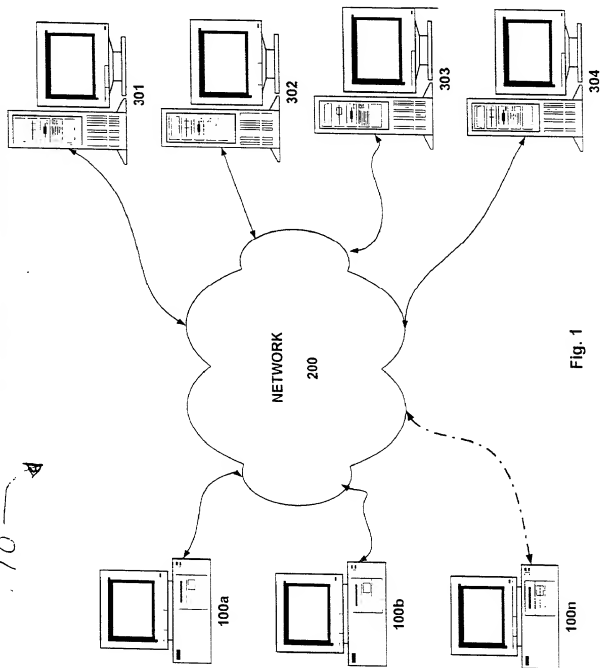


Fig. 1







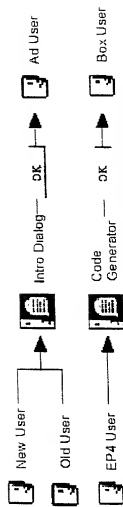


Fig. 4A

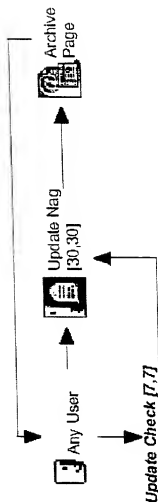


Fig. 7A

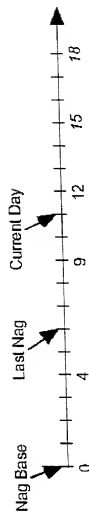


Fig. 11

### Welcome to Eudora!

Eudora is now licensed in three ways: Sponsored Mode, Paid Mode, and Light Mode. Unless you change modes, Eudora will run in Sponsored Mode, meaning it will display ads.

We have done our best to present the ads in a way that respects the work you do in email. By allowing Eudora to display ads, you get the full power of Eudora for free and we can still pay our bills.

If you decide the ads are not for you, you can change modes. Paid Mode shows no ads. Current Eudora Pro 4.x users will be able to upgrade to Paid Mode for free. Other users will be able to pay a license fee to go to Paid Mode. At this stage in testing, the machinery for Paid Mode is not fully tested, and Paid Mode is unavailable. Light Mode also shows no ads, but has many fewer features.

To switch forms of Eudora, please use the "Payment & Registration" item in the Help menu. To learn more about the three modes, click on the "Tell Me More" button below.

Tell me more

OK

Fig. 4B

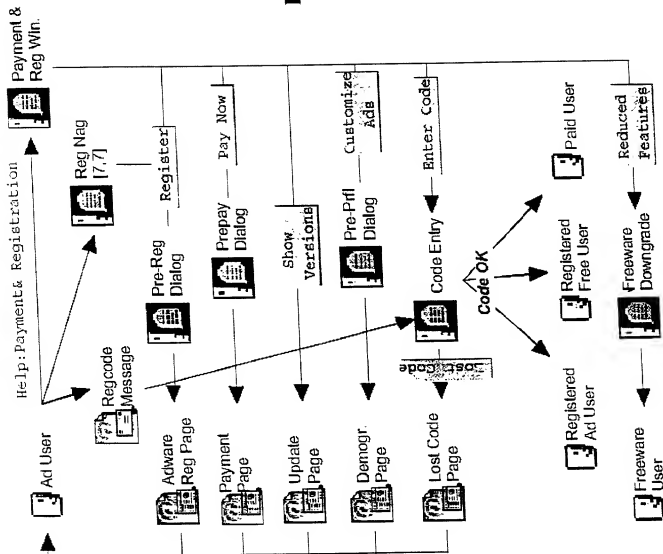


Fig. 5A





Would you like to register your copy of Eudora?

As a registered user of Eudora we won't nag you as often as we do. We'll also erect a giant statue in your image on the front lawn of our corporate headquarters (\*).

How cool is that? C'mon... register! It's fun and easy!

(\* Giant statue offer void on the planet Earth)

Maybe later

Take me to the registration page!

Fig. 5C

**Thanks for choosing to register Eudora!**

You'll next be walked through a few quick steps, as described below, before registration is complete:

- Eudora will open your web browser and take you to our registration page
- You'll fill in some simple registration information on the web site
- We'll then email a Eudora registration code back to you
- The next time you check mail, Eudora will automatically recognize this code and display a dialog box inviting you to confirm your registration information
- Ta da! You'll then become a registered user of Eudora... Thanks!

**Cancel****Continue****Fig. 5D**

**Thanks for choosing to purchase Eudora!**

You'll next be walked through a few quick steps, as described below, before your purchase is complete:

- Eudora will open your web browser and take you to our Payment & Registration page
- You'll be asked to provide your payment and registration information on the web site
- We'll then email a Eudora registration code back to you
- The next time you check mail, Eudora will automatically recognize this code and display a dialog box letting you to confirm your registration information
- Ta-da! You'll then become a Paid mode user. Congratulations!

Cancel

Continue

**Fig. 5E**

Thank you for your registration!

To complete your registration, please enter the name you  
under and your registration code below.

The exact name you registered under:

First Name:  Last Name:

Your registration code:

Fig. 5F

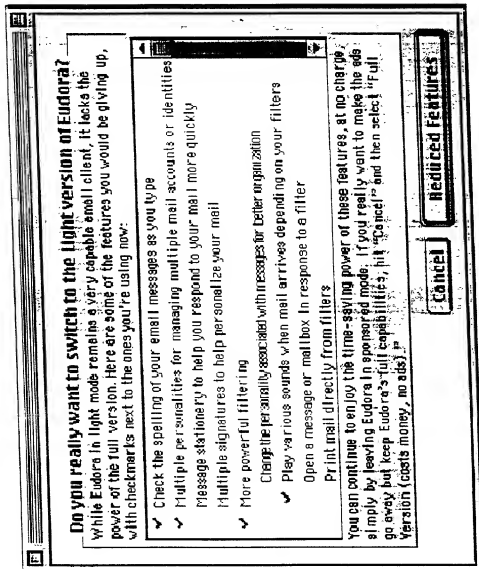
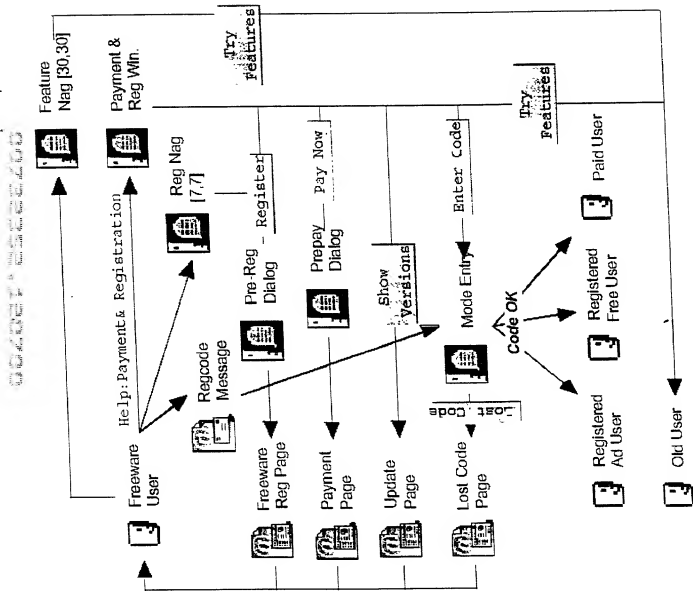


Fig. 5G



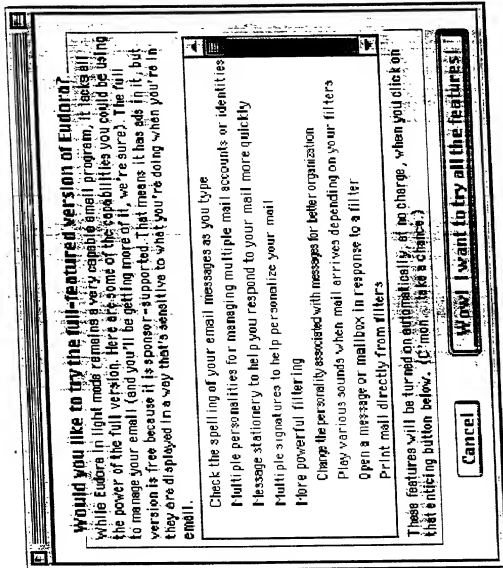
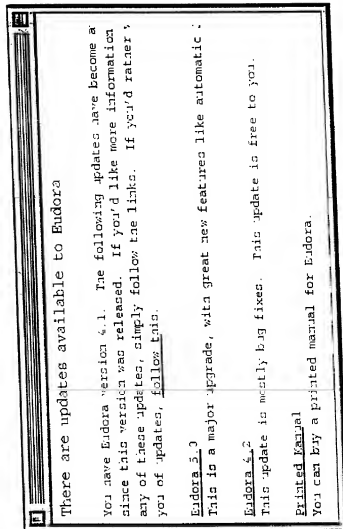


Fig. 6B



**Fig. 7B**



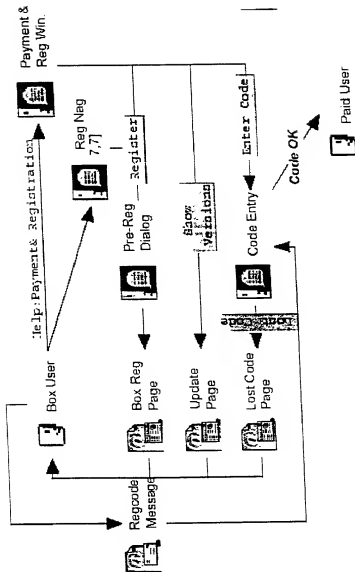


Fig. 8

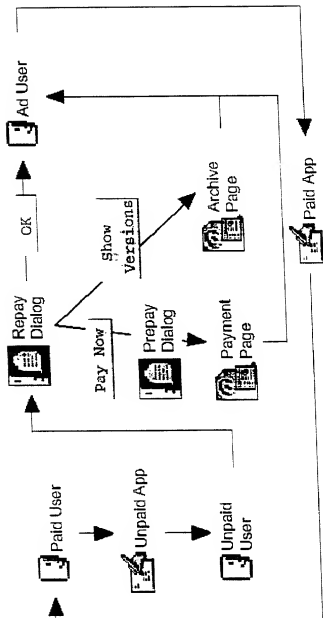


Fig. 9

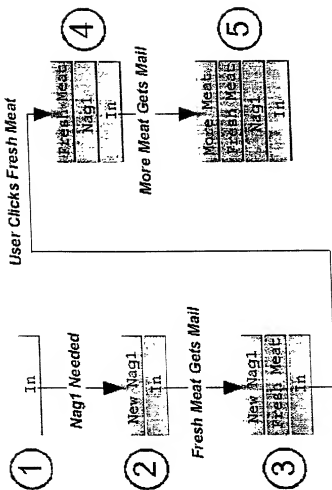


Fig. 10

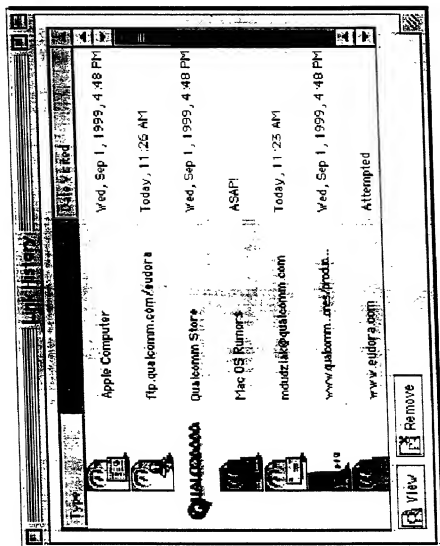


Fig. 12A

**You Can't Get There From Here**  
You're not connected to the Internet now. Help me cope.  
connect you and visit the site, record a bookmark for later  
remind you to visit it next time you are connected.

**Visit Now**

Connect to the Internet and visit the site

**Bookmark**

Bookmark this site to visit later

**Remind Me**

Bookmark the site, and remind you  
you're connected to the Internet

☐ Remember your choice for next time

**Fig. 12B**





```

////////////////////
// Main ad scheduler
ScheduleMain
{
    // Has a new day dawned?
    Do CheckForNewDay
    // Are we are within the current ad's showFor?
    if ( ad.thisShowTime < ad.showFor )
    {
        // there is nothing to be done
        return
    }
    // At this point, we know that we need a new ad
    // Perform housekeeping tasks on the old one
    Do AdEndBookkeeping
    // Pop out of a block if all ads on par
    if ( block isn't all playlists )
    {
        find ad with minimum ad.numberShown
        if ( ad.numberShown >= blockGoal )
            set block to all playlists
    }
    // If we are over our quota of regular ads for the day,
    // look for a runout
    if ( ad.faceTimeToday > faceTimeQuota )
    {
        Do ShowARunout
    }
    else
    {
        Do ShowARegularAd
    }
}
// end ad schedule main

```

Fig. 15A



```

////////////////////////////////////
// We must perform certain tasks when the calendar day
changes.
CheckForNewDay
{if ( the calendar day has changed )
{
// Perform housekeeping tasks on the ad currently showing
Do StopShowingCurrentAd
// Runout ads are charged for a full showFor if they've been
shown
// at all on a given day. Charge any runout ads if they've
been
// shown at all.
for runout ads
{
if ( ad.thisShowTime > 0 )
{
ad.totalTimeShown += ad.showFor
ad.thisShowTime = 0
}
}
// Now, reset the counters for all ads to reflect the fact
that
// a new day has dawned.
for all ads
{
ad.numbersShownToday = 0
}
// Record yesterday's facetime
// Might not literally be yesterday, be sure to use
// whatever day the app was last run on
set old current day's facetime to totalFaceTimeToday
// and reset our global regular ad facetime counter
adFaceTimeToday = 0
totalFaceTimeToday = 0
// if we were in a block, back out
set block to all playlists
}
}
// end CheckForNewDay

```

Fig. 15B

```

////////////////////////////////////
// This function shows a runout ad, and if it
// can't find one, goes to a rerun
ShowARunout
{
  for runout ads
  {
    // has the ad been flushed?
    if ( ad.flushed )
      try next ad
    // are we done showing this runout today?
    if ( ad.numberShownToday > ad.dayMax )
      try next ad // this one's used up for the day
    // are we done showing this runout for ever and ever?
    if ( ad.shownFor > ad.showForMax )
      try next runout ad // this one's used up forever
    // are we between the ad's start and end dates?
    if ( ad.startDate < the current date < ad.endDate )
      try next runout ad
    // the ad is not supposed to run today
    // do we actually HAVE the ad?
    if ( ad has not been downloaded )
    {
      ask for ad to be downloaded
      try next ad
    }
    // ok, we believe we should show this runout
    // we are now in runout state
    Do ShowAnAd
    return
  }
  // if we haven't found a runout ad, we will go to "rerun"
  state
  Do ShowARerun
}
// end ShowARunout

```

Fig. 15C

```

////////////////////////////////////
// Rerun state. Look for a regular ad to rerun
ShowARerun
{
  for regular ads [ in current block ]
  {
    // has the ad been flushed?
    if ( ad.flushed )
      try next ad
    // is this ad recent enough to rerun?
    if ( ad.lastShownDate is older than returnInterval )
      try next ad
    // this one is too old to rerun
    // if in block, show ads only if it's their "turn"
    if ( ad.numbersShownToday >= blockGoal )
      try next ad // need to find a friend in this block
    // are we between the ad's start and end dates?
    if ( ad.startDate < the current date < ad.endDate )
      try next ad
    // the ad is not supposed to run today
    // do we actually HAVE the ad?
    if ( ad has not been downloaded )
    {
      ask for ad to be downloaded
      try next ad
    }
    // ok, at this point we can show this ad, but because
    // we're in rerun, we don't keep the books
    Do ShowAnAd
    return
  }
  // if we get here, we have no ads to show. Punt.
  return
}
// end ShowARerun

```

Fig. 15D

```

////////////////////////////////////
// Show a regular ad
ShowARegularAd
{
for regular ads [ in current block ]
{
// has the ad been flushed?
if ( ad.flushed )
try next ad
// are we done showing this ad today?
if ( ad.numberShownToday > ad.dayMax )
try next ad // this one's used up for the day
// if in block, show ads only if it's their "turn"
if ( ad.numberShownToday >= blockGoal )
try next ad // need to find a friend in this block
// are we done showing this ad for ever and ever?
if ( ad.shownFor > ad.showForMax )
try next ad // this one's used up forever
// are we between the ad's start and end dates?
if ( ad.startDate < the current date < ad.endDate )
try next ad
// the ad is not supposed to run today
// do we actually HAVE the ad?
if ( ad has not been downloaded )
{
ask for ad to be downloaded
try next ad
}
// ok, we believe we should show this ad
// we are now in regular state
Do ShowAnAd
return
}
// If we get here, we have failed to find a regular
// ad. Go to runout
Do ShowARunout
}
// end ShowARegularAd

```

Fig. 15E

```

////////////////////////////////////
// Perform necessary housekeeping when we're taking
// down an ad
AdEndBookkeeping
{
// In rerun state, we don't do any bookkeeping
if ( in RerunState )
return
// Account for at most ad.showFor seconds, provided
// we've shown the ad for at least ad.showFor seconds
// Note that this means we don't charge for time beyond
// ad.showFor seconds, which is important
if ( ad.thisShowTime >= ad.showFor )
{
ad.numberShownToday += ad.showFor
ad.shownFor++
// we do NOT reset thisShowTime here, we do it in
// AdStartBookkeeping. It actually doesn't matter where
// we do it, provided we are careful NOT to do it for
// runout ads.
}
}
// end AdEndBookkeeping

```

Fig. 15F

```

////////////////////////////////////
// Show an ad, including bookkeeping and block handling
ShowAnAd
{
  // If the ad is in a block, notice that
  if ( it's in a "block" playlist )
  {
    if ( not currently in a block )
    {
      find ad in block with minimum numbersShown
      make that our ad
      set blockGoal to minimum numbersShown+1
    }
    set current block to this playlist
  }
  // now do bookkeeping
  Do AdStartBookkeeping
  // and actually show it
  Do DisplayThatAd
}

```

Fig. 15G

```

////////////////////////////////////
// Perform housekeeping when we put up an ad
AdStartBookkeeping
{
  // In rerun state, we don't do any bookkeeping
  if ( in RerunState )
    return
  // For regular ads
  if ( it's a regular ad )
  {
    ad.thisShowTime = 0
    ad.lastShownDate = now
  }
}
// end AdStartBookkeeping

```

Fig. 15H

Persistent Ads	
PlayList Request	faceTime Used to determine how much advertising to send to client faceTimeLeft Not used
PlayList Response ClientInfo	reqInterval Relatively large, one or more days flush Used Single playlist completely specifies list of ads client should have
PlayList Response Scheduling Parameters	showForMax Not used

Fig. 16A

Short-Lived Ads	
PlayList Request	faceTime Not used faceTimeLeft Used to determine how many ads client should receive
PlayList Response ClientInfo	reqInterval Not used Instead client requests new playlist whenever ads "run low". flush Not used
PlayList Response Scheduling Parameters	showForMax Used to determine how long an ad runs

Fig. 16B



**Eudora doesn't seem to be getting ads.**  
For some reason, Eudora is unable to download new ads. Downloading and displaying ads is a requirement for the free full-featured version of Eudora. Please visit the Eudora web site for information about how to resume getting ads.

Invalid HTTP request (Error code: 503)

**If ad downloading continues to fail, Eudora will eventually revert to the Light version which is less powerful.**

[Take me to the Eudora web site](#)

Fig. 17A

**Something seems to be covering the ad.**



It's probably inadvertent, but Eudora has determined that you are covering up all or a significant portion of an ad. The software is designed to notify you when this happens in the hopes that you will stop covering up the ad. If you don't, this window will keep popping up (which you will probably find quite annoying).

We've always got some good stuff under development back at the home office, and it's the advertising in Eudora that enables us to continue to develop the software while providing it to you for free. We've worked hard to make sure the advertising isn't annoying and we genuinely hope that you are not deliberately trying to cover the ads because they're bothering you. Of course, you can choose to pay us for Eudora by choosing "Payment & Registration" from the "Help" menu and clicking on "Paid Full Version." Or you can remove whatever is obscuring the ad.

OK

**Fig. 17B**

002007-03000240

**Eudora will now revert to a less powerful version.**  
Eudora has been unable to download ads for quite some time and will now  
revert to a less powerful version. If you would like more information about  
why Eudora's features are being reduced at this time, please visit the Eudora  
web site. You will find information there about how the full-featured  
version can be reactivated.

We're sorry for this inconvenience.

[Take me to the Eudora web site](#)

Sadly, OK

Fig. 17C

### We'd like to know how you use Eudora.

In order to make Eudora work as well as possible, it's important that we know how people use it. We ask users for this information at random. Looks like it's your turn. If you're open to helping us this way, all you have to do is click "Generate Info" below and a message will be created. You can review the contents of the message if you like, and then send it to us or not --- that's up to you.

We value our privacy; we're pretty sure you value yours. So we want you to know what we'll be collecting and give you a chance to eliminate anything you don't want to send. Simply uncheck the boxes next to any information you'd rather not send.

Please understand that as soon as we receive your email, we will throw away the headers that identify the mail as coming from you. You see, we don't actually need to know who you are to find your information helpful. So we promise to protect your privacy and turn you into "just a number."

### It's OK to transmit statistics regarding:

- ☒ Your demographic data
- ☒ Your key/Eudora usage
- ☒ Advertisement information
- ☒ Eudora features you use
- ☒ Non-personal settings

Cancel

Generate Info

Fig. 18A

Page	Applicable Query Parts											topic			
	platform	product	version	distribut	mode	realname	email	regfirst	reglast	regcode	oldReg		regLevel	profile	url
Payment	action	X	X	X	X	X	X	X	X	X	X	X			
Freeware Registration	pay	X	X	X	X	X	X	X	X	X	X	X			
Adware Registration	register-free	X	X	X	X	X	X	X	X	X	X	X			
Box Registrations	register-ad	X	X	X	X	X	X	X	X	X	X	X			
Lost Code	register-box	X	X	X	X	X	X	X	X	X	X	X			
Update	lostcode	X	X	X	X	X	X	X	X	X	X	X			
Pro Update	update	X	X	X	X	X	X	X	X	X	X	X			
Archived	pronupdate	X	X	X	X	X	X	X	X	X	X	X			
Profile	archived	X	X	X	X	X	X	X	X	X	X	X			
Introduction	profile	X	X	X	X	X	X	X	X	X	X	X			
Support	intro	X	X	X	X	X	X	X	X	X	X	X			
QuickTime Missing	n/a	X	X	X	X	X	X	X	X	X	X	X			no-qt
Ad Failure	support	X	X	X	X	X	X	X	X	X	X	X			ad-fall
Tutorial	support	X	X	X	X	X	X	X	X	X	X	X			tutor
FAQ	support	X	X	X	X	X	X	X	X	X	X	X			faq
Light Users	support	X	X	X	X	X	X	X	X	X	X	X			light
Search Support	support	X	X	X	X	X	X	X	X	X	X	X			search
NewsGroups	support	X	X	X	X	X	X	X	X	X	X	X			usenet

Fig. 19

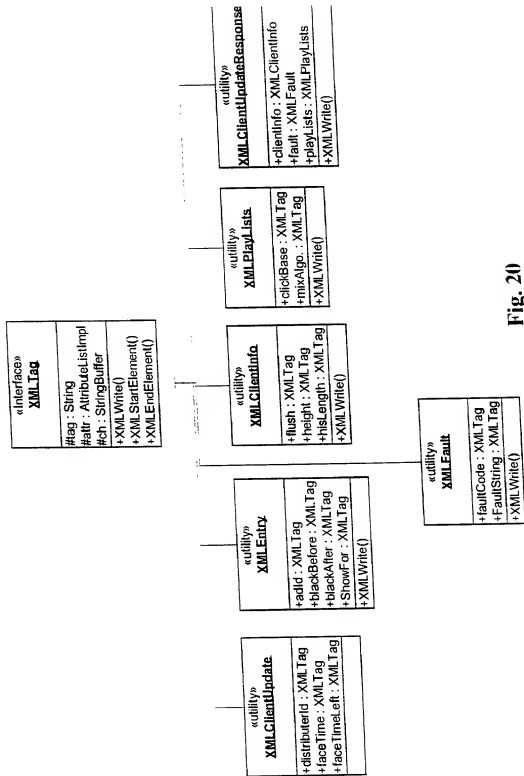


Fig. 20

```

8 The list of available ads advantageously can be built from the following query:

ads = dbCon.prepareStatement("SELECT * FROM ads WHERE StartDate <= today AND endDate >= today + 30 AND
AdType = 'a' AND AdStatus = 'A' AND ImpressionsServed < Impressions ORDER BY ImpressionsServed ASC");

run out ads = dbCon.prepareStatement("SELECT * FROM ads WHERE StartDate <= today AND endDate >= today +
30 AND AdType = 'R' AND AdStatus = 'A' AND ImpressionsServed < Impressions ORDER BY ImpressionsServed
ASC);

8 The time required to deliver the ads advantageously can be calculated in the following manner.

face time left for today [seconds] = faceTime[today] - faceTimeUsedToday

(Comment: Face time left for today is the number of seconds the servlet can use to deliver special ads today.)

predictFaceTime [seconds] = SUM( faceTime[tomorrow], faceTime[tomorrow + 1], ..., faceTime[tomorrow + reqInterval]
)

(Comment: Predict face time is the number of seconds the servlet predicts the user is going to have.)

goalShowTimeLeft [seconds] = predictFaceTime - faceTimeLeft

(Comment: Goal show time left is the number of seconds that the software provider needs to fill with ads.)

```

Fig. 21A

```

* Targeting
  while (face time left for today ) {
    if ad is not in the history {
      select ad [according to target = today]
      face time left for today := ad.showFor
    }
    next ad
  }

  while (Goal show time left ) {
    if ad is not in the history {
      select ad [according to target]
      goal show time left := ad.showFor
    }
    next ad
  }

```

Default values:

```

reqInterval = 1 day,
faceTime = 30 minutes
faceTimeQuota is ?
hisTimeLength = 31 days

```

Fig. 21B



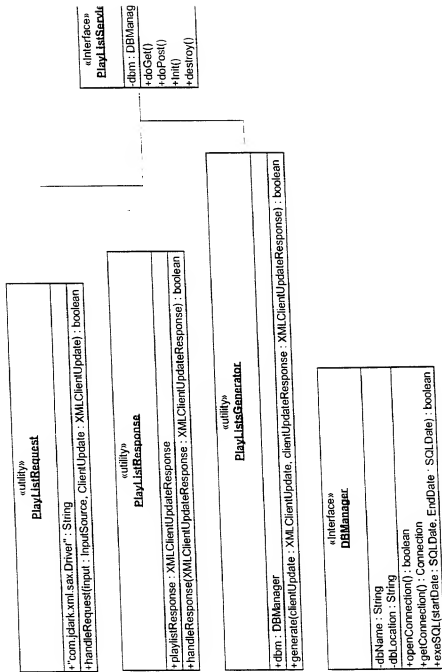
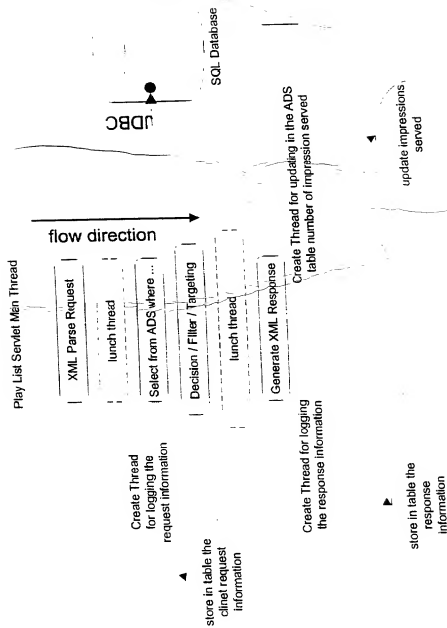


Fig. 22



**Fig. 23**